Computational Complexity and the Cook-Levin Theorem

Paul Sacawa

MAT 477

February 11, 2014

Our Goals from a Naive Perspective

A Turing Machine (shortly TM) is an algorithm which takes an input and either solves a decision problem (answering 'yes' or 'no', depending on the input) or computes a function. The class **P** contains decision problems which can be solved in a number of steps polynomial in the size of the input (shortly, by a polytime algorithm). The class **NP** contains problems for which to a 'yes' response can be given a proof verifiable by means of polytime TM (shortly, certified in polytime). Therefore, $\mathbf{P} \subset \mathbf{NP}$.

A Million Dollar Problem: Is $\mathbf{P} = \mathbf{NP}$?

In other words, if there is a proof verifiable in polytime that a property holds (**NP**), could we have computed in polytime whether it holds (**P**), i.e. perhaps without a proof ?

We say a decision problem A is reducible to a decision problem B if a polytime TM computes a function f translating inputs of A into inputs of B in a way that preserves the response 'yes' or 'no'. We write $A \leq_p B$.

If $B \geq_p NP$, $NP \neq P$ then B is not polytime $\Rightarrow$ impractical to compute.

NPH $\equiv$ **NP-hard** are the problems that all of **NP** are reducible to,

i.e. shortly, problems at least as hard as each problem of **NP**.

Finally, **NPC** $\equiv$ **NP-complete** $=$ **NP-hard** $\cap$ **NP**, i.e. **NP** problems that

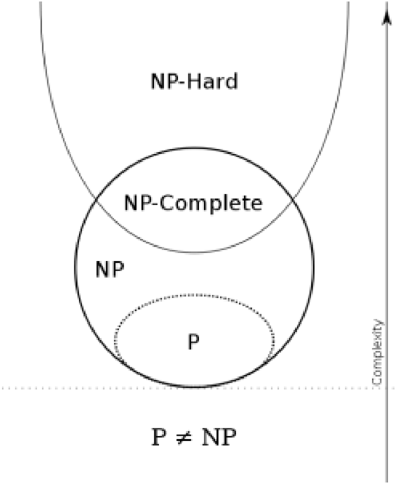
are the hardest in all of **NP** in the sense of the $\leq_p$ ordering. It is possible

that the set of maximums of this very broad class **NP** in terms of

computational difficulty is empty. But, we'll prove otherwise:

The big picture

The following picture (dependent on the unproven hypothesis $\mathbf{P} \neq \mathbf{NP}$) shows the relation of the complexity classes discussed.



Def : SAT is the problem of determining, given Φ a formula built from variables $Var = \{v_1, v_2, v_3 \ldots\}$ and connectives $\vee :=$ or, $\wedge :=$ and, $\neg :=$ negation, if there is a truth assignment $\tau : Var \to \{True, False\}$ that makes $\Phi[\tau]$ true. (shortly, 'satisfying' truth assignment).

Main Theorem (Cook, Levin) SAT $\in$ **NPC**, so **NPC** is nonempty.

Remark. NP $\ni$ **SAT** $:= \{\langle\Phi\rangle : \Phi$ is a satisfiable sentential formula$\}$, where $\langle\Phi\rangle$ is a string in Σ^* representing here Φ, or later other data.

Proof. A formula Φ can be certified as satisfiable by giving a truth assignment that makes it true. So, let the machine V take some

assignment τ and verifiy whether $\Phi[\tau]$ is true. **It is a fact in the subject**

that in the formal model of Turing Machines this can be done in polytime

and the proof has polynomial length. $\blacksquare$.

So, to prove **SAT** $\in$ **NPC** it suffices to show $A \leq_p$ **SAT** for any $A \in$ **NP** .

This means that given a polytime verifier machine V for A , we need to

make a polytime computable translation $f : \Sigma^* \to \Sigma^*$ from A to **SAT** (in

terms of strings of Σ^* of symbols in Σ) satisfying $x \in A \iff f(x) \in$ **SAT**

Rigor begins here : Claim. **SAT** $\in$ **NP-hard**.

Proof. For any $A \in$ **NP** there is a polytime machine V and $j \in \mathbb{N}$ with

$x := x_1 x_2 \dots x_n \in A$ iff $\exists$ certificate of "true" $y \in \Sigma^*$ satisfying $|y| \leq |x|^j$

such that $V(x, y)$ accepts. So, given x , we must exhibit a formula Φ_x for

which assignments of its variables indicate possible computations of V ,

and a satisfying assignment corresponds exactly to an accepting

computation. This $f : x \mapsto \Phi_x$ will be our reduction showing $A \leq_p$ **SAT** .

With n the size of the input and $p(n) \in \mathbb{N}$ running time of V (# of steps)

we construct f starting with variables appearing in Φ_x :

For each $i, j \leq p(n)$ and $\lambda \in \Sigma$ we consider a sentential variable $T_{ij\lambda}$

representing $T_{ij} = \lambda$, namely: the j^{th} tape cell at the i^{th} step has $\lambda \in \Gamma$.

For each $i \leq p(n)$ and state $q \in Q$ we consider a sentential variable Q_{iq}

representings that at the i^{th} step of the computation our V is in state q .

For each $i, j \leq p(n)$ we consider a variable H_{ij} representing that at the

i^{th} step of the computation, the tape head is at the j^{th} cell.

The Structure of Φ_x :

We construct our formula Φ_x on the variables $T_{ij\lambda}$, Q_{iq} , H_{ij} as

$$\Phi_x = \Phi_{initial} \wedge \Phi_{final} \wedge \Phi_{unique} \wedge \Phi_{compute} \quad , \textit{ where}$$

- $\Phi_{initial}$ asserts that the machine is appropriately set in the first step of computation: $\langle x, \cdot \rangle$ is on the tape and the state is q_0 , etc.
- Φ_{final} asserts that the computation of V on input $\langle x, \cdot \rangle$ accepts (here our final state is q_{accept}).

- Φ_{unique} asserts that we have not set the values of the variables inconsistently, i.e. for each i, j we can assign "true" only to one $T_{ij\lambda}$, since the tape cell j at any step i has only one value.
- $\Phi_{compute}$ asserts that the assignments of the tape cells follow in accordance with the transition function δ in order that the assignment of the variables will represent a valid computation.

Construction of $\Phi_{initial}$.

Initially we need the tape contents to be $\langle x, y \rangle$ for arbitrary y , the initial state to be q_0 and the head of the tape to be at cell 1 . So we set

$$\Phi_{initial} := \bigwedge_{i=1}^{r} T_{i1x_i} \wedge Q_{1q_0} \wedge H_{11}$$

The first part sets the first characters of the input to $x = x_1 x_2 \dots x_r$, the second part forces the first state to be q_0 , and the third forces the tape head to be at the first cell.

So, the length of the formula $\Phi_{initial}$ is bounded by $O(n)$.

Construction of Φ_{final} .

Φ_{final} encodes the appropriate ending conditions. Since we want

$V(x, y)$ to accept, we just need the final $p(n)^{th}$ step of the

computation to be in state q_{accept} . Therefore we set

$$\Phi_{final} := Q_{p(r)\ q_{accept}}\ ,\ for\ r = n\ .$$

So, the length of the formula Φ_{final} is bounded by $O(1)$.

Construction of Φ_{unique} .

Here we simply must ensure that tape cells will not have simultaneously multiple values of symbols and that at each step our TM has a unique state and a unique position of the tape head. Therfore we set

$$\Phi_{unique} := \bigwedge_{i,j\leq p(n)} \bigwedge_{\lambda\in\Gamma} \bigwedge_{\kappa\in\Gamma\setminus\lambda} (T_{ij\lambda} \to \neg T_{ij\kappa}) \wedge \bigwedge_{i\leq p(n)} \bigwedge_{q_1,q_2\in Q} (Q_{iq_1} \to \neg Q_{iq_2}) \wedge \bigwedge_{i\leq p(n)} \bigwedge_{j_1,j_2\leq p(n)} (H_{ij_1} \to \neg H_{ij_2})$$

The lengths of the 1^{st} , 2^{nd} and 3^{rd} blocks of the formula Φ_{unique} are bounded by $O(p(n)^2)$, $O(p(n))$ and $O(p(n)^3)$.

This forces an assignment satisfying Φ_{unique} to generate at every step of

our TM a choice of the tape contents, the state, and the head position.

Our formula says that for each step of our TM "true" values of $T_{ij\lambda}$,

Q_{iq} , H_{ij} are set to be unique, i.e. "false" is set for any other values

of the secondary variables.

So, the length of the formula Φ_{unique} is bounded by $O(p(n)^3)$.

Construction of $\Phi_{compute}$ of the size $O(p(n)^2)$.

Most important and difficult is to demonstrate how $\Phi_{compute}$ relates the states to the $T_{ij\lambda}$, Q_{iq} , H_{ij} in correspondence with the code of the TM expressed by means of the transition function δ . We express $T_{ij\lambda}$, Q_{iq} , H_{ij} as boolean functions of the same (for whatever i) number of variables $T_{i-1\ j\kappa}$, $Q_{i-1\ q}$ and $H_{i-1\ j}$; and consequently, as an $O(1)$ size formula. We then express $\Phi_{compute}$ as the conjunction of all of them over all tape cells and steps of our TM. Then the length of $\Phi_{compute}$ is $O(p(n)^2)$.

$\Phi_x \in \textbf{SAT} \iff \exists\, y : V(x,y)$ accepts:

Based on the construction of the formula Φ_x , it follows that a

"satisfying" assignment of $T_{ij\lambda}$, Q_{iq} , H_{ij} corresponds exactly to a

choice of $\langle x, y \rangle$ in the tape cells $T_{1j\lambda}$ that represent the tape in the

1^{st} step of our TM and the following this step consistent computation

that ends in "acceptance". Such string $y \in \Sigma^*$ exists iff $x \in A$

because of the "verifier" role of our Turing Machine V , i.e.

$$x \in A \iff \exists\, |y| \leq |x|^k : V(x,y) \text{ accepts} \iff \Phi_x \text{ satisfiable} .$$

End of Proof.

Moreover, formula Φ_x has size $O(p(n)^3)$. Consequently,

constructed function $f_A : x \to \Phi_x$ is polytime computable and

$$x \in A \iff f_A(x) = \Phi_x \in \mathbf{SAT} .$$

Therefore $A \leq_p \mathbf{SAT}$ for arbitrary $A \in \mathbf{NP}$, i.e. **SAT** is in **NPC** ,

as required. ■

If $\mathbf{P} \neq \mathbf{NP}$ then $\mathbf{NPC} \Rightarrow$ impractical

Def. Say a sentential logic formula Φ is in 3-CNF form if it has the form

$\Phi = (c_{11} \vee c_{12} \vee c_{13}) \wedge \cdots (c_{k1} \vee c_{k2} \vee c_{k3}) \wedge \cdots \wedge (c_{n1} \vee c_{n2} \vee c_{n3})$, where

each c_{jk} is either x or $\neg x$, for a variable x.

Def. **3-SAT** $= \{\langle \Phi \rangle : \Phi$ is a satisfiable 3-CNF formula$\}$

Thm. (left without proof) **3-SAT** $\in$ **NPC** .

In a graph G, a k-clique is a subset C of k vertices which are all

connected to each other by edges.

Thm. **CLIQUE** $:= \{\langle G, k\rangle : G$ has a k-clique$\} \in$ **NPC** .

Proof. **CLIQUE** $\in$ **NP** (the clique can be the certificate), so it suffices to

show **3-SAT** $\leq_p$ **CLIQUE**. Then, given $\Phi = \bigwedge_{j=1}^{r}(c_{j1} \vee c_{j2} \vee c_{j3})$

o 3-CNF formula n variables $x_1, \ldots, x_n$, we set $k = r$ and consider

graph $G_\Phi = (V_\Phi, E_\Phi)$ with vertex set

$V_\Phi = \{(\sigma, i) :$ σ is either x_m or $\neg x_m$ and σ appears as some $c_{ik}\}$ and

$E_\Phi = \{((\sigma, i), (\delta, j)) : i \neq j$ and $\sigma \neq \neg\delta\}$ Then a clique in graph G_Φ is

exactly a choice of k values for c_{ij}, one for each triple, which sets Φ true.